Inference in Directed Acyclic Graphs with Applications to Hidden Markov Model Structures

Padhraic Smyth[+], David **Heckerman***, Paul **Stolorz**[+]
[+]Jet Propulsion Laboratory, MS 525-3660
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

*Microsoft Research, Bldg 9S
Redmond WA, 98052-6399

Abstract

Graphical techniques for modeling the dependencies of random variables have been explored in a variety of different areas including statistics, statistical physics, artificial intelligence (AI), speech recognition, image processing, and genetics. Formalisms for manipulating these models have been developed relatively independently in these research communities. In this paper we explore hidden Markov models (HMMs) and related graphical structures within the general framework of directed acyclic graphs (DAGs). In particular we show that there exists a general methodology for deriving an exact inference method for any DAG, that this algorithm produces the well-known forward-backward and Viterbi algorithms as special cases, and that the complexity of inference can be characterized for general families of DAGs including HMMs which are augmented with non-local dependencies. Applications to problems such as speech recognition and biological sequence modeling are discussed.

1 Introduction

A directed acyclic graph (DAG) is a graphical specification of the dependencies which exist among a set of random variables, such that each node corresponds to a random variable and each directed link corresponds to a statement that the probability of the node at the tail of the link is dependent on the node at the head of the link. In this paper, we are concerned with *inference* problems in DAGs: the calculation of posterior probabilities of variables of interest given observable data and/or a specification of the graphical model, and the related task of MAP identification the determination of the most likely state of unobserved variables, given observed variables. The learning or estimation problem, where one determines the parameters (and possibly structure) of the model from data is not addressed in any depth. Nonetheless, because inference is usually an "inner loop" in the learning process (for example in expectation-maxi]) ization type algorithms), specification Of

inference algorithms and results on their complexity have a direct bearing on the learning problem.

We examine a particular inference algorithm for DAGs with arbitrary structures. The probabilities calculated by the algorithm are *exact* and not approximations. This algorithm, developed by Jensen, Lauritzen and Oleson (1989), hereafter referred to as the JLO algorithm, is a descendant of a Bayesian-network inference algorithm first described by Lauritzen and Spiegelhalter (1988). A closely related algorithm, developed by Dawid (1992), solves the MAP identification problem with the same time complexity as JLO'S inference algorithm. Although the algorithms are generic, we show that their application to standard problems, such as the HMM, yields optimal or near-optimal asymptotic performance. When applied to an HMM, for example, these algorithms recreate the well-known forward-backward and Viterbi algorithms in a more general context.

Furthermore, we use the inference algorithm to derive the inference complexity for a particular class of graphical model structures-namely, that of an underlying first-order Markov (or hidden Markov) chain that also has some non-local dependencies between the variables. Non-local dependency models of this sort are particularly suitable for practical problems where the physical structure of the problem dictates both a local Markov dependency and longer-range constraints (such as biological sequence modeling). One of the primary results of the paper is the identification of classes of non-local models where the complexity of inference scales linearly *rather than exponentially* in the *distance* of the non-local dependencies.

2 Inference in Arbitrary Directed Acyclic Graphs

In this section, we describe a special case of the inference algorithm for DAGs described by JLO, and mention the corresponding algorithm for MAP identification developed by Dawid (1992).

First, let us examine the special case of the inference problem, where we are given a DAG and want to compute the probability distribution of a set of observations O' for the variables O in the domain of the DAG. We illustrate the JLO algorithm for the simple DAG over discrete variables $U = \{x_1, \ldots, x_6\}$ shown in Figure 1a. The algorithm performs inference in five steps: moralization, triangulation, clique-tree formation, initialization, and propagation. The first four steps are carried out only once for a given DAG. The propagation step is carried out each time a new inference for the given DAG is requested.

Moralizing a directed graph consists of adding undirected links between unconnected parents of common children and then dropping the directionality on all links: the resulting non-directed graph is known as a *moral graph*. Figure 1b is the moral graph for Figure 1a. A graph is *triangulated* if every cycle of length greater than three has a chord. In the second step of the algorithm, links are added to the moral graph of the given DAG until it is triangulated. We describe this step in more detail later in this section. A triangulation of the DAG in Figure 1b is shown in Figure 1c.

The triangulation of the moral graph of the DAG defines a set of cliques. maximal sets of variables that are all pairwise linked. A *clique tree* is a tree where each node corresponds to a particular clique obtained from the DAG triangulation, and arcs connect nodes so as to guarantee the running intersection *property*: if the same variable is present in two different cliques of the clique tree, then that variable occurs in all cliques along the path between the two original cliques. In the third step of the algorithm, we construct a clique tree from the DAG triangulation, and (arbitrarily) choose a root for the tree. We describe the construction of a clique tree in more detail shortly. A clique tree corresponding to the cliques in Figure 1c is shown in Figure 1d. Each node in the clique tree is initialized to the joint probability distribution of the variables in that node, as determined by the original DAG, using calibration operations as described below.

The *propagation* step of the algorithm itself has two phases: instantiate and collect. In the *instantiation phase*, for each evidence variable that has been observed, we find a clique (ideally, the smallest clique) containing that variable, and set to zero the probability of any state in the joint distribution of the marginal that is inconsistent with the observation. At this point, we do not renormalize the probabilities in the clique, although we continue to refer to them as probabilities.

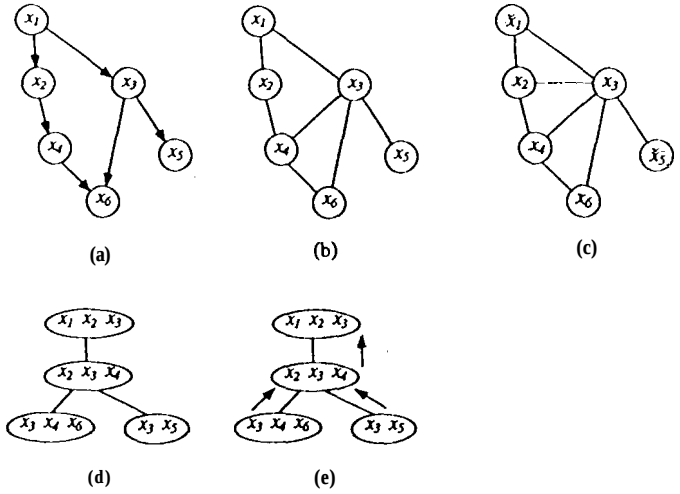

Figure 1: (a) A simple directed acyclic graph. (b) The corresponding (undirected) moral graph. (c) The corresponding triangulated graph. (d) The corresponding clique tree. (e) The calibration step when the top node is taken as the root.

In the *collect phase,* messages are passed between variables so aa to *calibrate* the probabilities of the cliques based on the changes made during initialization. In particular, calibrations proceed from the leaves of the clique tree to its root as shown in Figure le.

To understand how to do calibration, suppose the probabilities of clique C_1 have changed to $p^*(CI)$, either due to instantiation or to a previous calibration. Further, suppose we want to calibrate the neighbor of C_1 in the clique tree, say C_2. Let Xl, X_2, and I_{12} denote the variables unique to C_1, the variables unique to C_2, and the intersection of C_1 and C_2, respectively. That is, $C_1 = X_1 \cup I_{12}$, $C_2 = X_2 \cup 1_{12}$, and Xl, X_2, and I_{12} are disjoint. First, we marginalize the new distribution of Cl to obtain the new distribution of I_{12}:

$$p^*(I_{12}) = \sum_{X_1} p^*(I_{12}, X_1) = \sum_{X_1} p^*(C_1)$$

Then, we absorb the update into the joint distribution of C_2 using the relation

$$p^*(C_2) = p^*(X_2, I_{12}) = p(X_2|I_{12})\, p^*(I_{12}) = p(C_2)\, \frac{p^*(I_{12})}{\sim J(\sim\sim 2j-}$$

During the collect phase, if a node has more than one child, the node is calibrated sequentially by each child.

It is not difficult to show that, after a clique C has been calibrated, p^* (C) is equal to $P(C, O_b = O'_b)$, where O_b is the subset of observations in O at or below C in the clique tree. Consequently, we can obtain the probability of interest by summing the calibrated probabilities of the root clique C_r over all states of C.:

$$p(O = o') = \sum_{C_r} p^*(C_r)$$

The time complexity of propagation within the clique tree G_T is

$$C(G_T) = 2S(G_T) = 2 \sum_{i=1}^{N_C} \text{size}(C_i)$$

where $S(G_T)$ is the sum over all cliques of the number of states in each clique, N_C is the number of cliques in the tree, and $\text{size}(C_i)$ is the number of states in the joint space of C_i (equal to the product over each variable in C_i of the number of states of each variable). Thus, for inference to be efficient, we need to construct clique trees with small clique sizes. Problems of finding optimally small clique trees (e.g., finding the clique tree with the smallest maximal clique) are NP-hard. Nonetheless, Jensen et al. (1990) and other researchers have identified a collection of good heuristics for constructing clique trees from DAGs.

A simple greedy algorithm for triangulation is based on the fact that a graph is triangulated if and only if all of its nodes can be eliminated, where a node can be eliminated whenever all of its neighbors are pairwise linked. Whenever a node is eliminated, it and its neighbors define a clique in the clique tree that is eventually constructed. Thus, we can triangulate a graph and generate the cliques for the clique tree by eliminating nodes in some order, adding links if necessary. If no node can be eliminated without adding links, then we choose the node that can be eliminated by adding the smallest number of links. If more than one node can be eliminated, we choose the one that creates the smallest clique. The links added by this algorithm applied to the graph in Figure 1b are shown in Figure 1c.

A simple greedy algorithm for clique-tree construction is based on the following fact. Define the weight of a link between two cliques and the number of variables in their intersection. Then, a tree of cliques will satisfy the running intersection property if and only if it is a spanning tree of maximal weight. Thus, we can construct a clique tree by choosing successively a link of maximal weight unless it creates a cycle. The clique tree constructed from the cliques defined by the DAG triangulation in Figure 1c is shown in Figure 1d.

As we mentioned in the introduction, the Dawid algorithm for MAP identification is analogous to the JLO inference algorithm. Namely, the same clique tree is used, and the propagation step is replaced by a dynamic program. The time complexity of MAP identification is the same as that of the inference.

3 Inference Applications

In this section, we consider a few real-world inference problems, and show that the JLO algorithm combined with the simple heuristics for clique-tree construction can provide inference algorithms for HMMs and related structures which are optimal or near-optimal in time complexity. Clearly there are many other interesting graph structures used in applications such as signature analysis, time-series modeling, etc., which are not discussed here.

3.1 Hidden Markov Models

Figure 2a shows a hidden Markov model (HMM) represented as a DAG. The hidden and observable nodes are $h_1, \ldots, h_N$ and $o_1, \ldots, o_N$, respectively. Figure 2b shows the corresponding (unique) clique tree constructed by the heuristics described in the previous section. Assuming that each node in the HMM has m states, the complexity of inference using JLO propagation is $O(Nm^2)$, which is asymptotically optimal. Furthermore, if we choose the clique $\{h_{N-1}, h_N\}$ as the root of the clique tree, then the collect steps carried out in the proper order correspond directly to the operations in the well-known forward-backward algorithm. Furthermore, Dawid's MAP identification algorithm applied to this clique tree corresponds to the Viterbi algorithm. (Formal proofs of these equivalences are omitted due to space). Note that both algorithms are clearly instances of dynamic programming, but that the Viterbi algorithm as used in HMM inference can be considered a special case of the Dawid algorithm which works for arbitrary graphs. Note that the general equivalence of DAGs and HMMs has been noted by Buntine (1994) and Frasconi and Bengio (1994) among others, although the demonstration of equivalence of specific inference algorithms is new as far as we are aware.

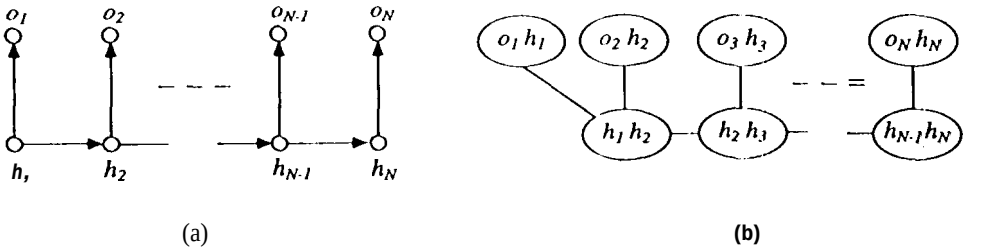

Figure 2: (a) A hidden Markov model. (b) A corresponding clique tree.

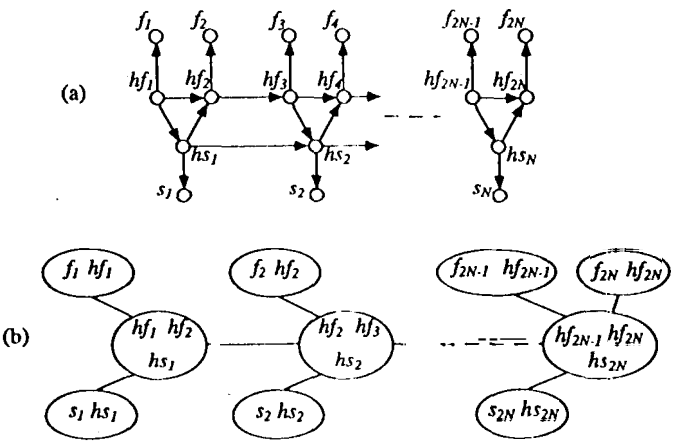

Figure 3: (a) A modified hidden Markov model for speech recognition. (A directed version of the graph in Saul and Jordan [1994]). (b) A corresponding clique tree.

3.2 Modified **HMMs** for Speech Modeling

Figure 3a shows a modified hidden Markov model for speech recognition: the undirected version of this graph was introduced by Saul and Jordan (1994) . The model is modified so as to handle feature sets with disparate time scales-in this case, a 2:1 disparity. The f_i and hf_i are the "fast" observables and hiddens, the s_i and hs_i the corresponding slow ones. Figure 3b shows a corresponding clique tree, again constructed by the algorithm described in the previous section. Assuming that each node in the HMM has m states, the complexity of inference using JLO propagation is $O(Nm^3)$, which is the same complexity as an optimal inference algorithm. Although the clique tree in Figure 3b is not unique, all clique trees for this DAG yield the same inference complexity.

3.3 Modeling of Biological Sequences

DAGs in general are applicable to biological sequence modeling problems such as the alignment of globular proteins. The standard HMM structure Figure 2a has previously been exploited by several groups (e.g., Baldi et al, 1994) to perform protein alignment in a way that captures first-order sequence information. In the context of protein alignment, the hidden nodes hi represent so-called insert and *delete* states, while the o_i represent the *match nodes* giving the probability of occurrence of each amino acid at each location in the protein sequence.

Figure 4a represents a generalization to include the case of a modest number of overlapping long-range dependencies. Node o_{k+1} has a non-local dependency on observable node 0_i, and observable

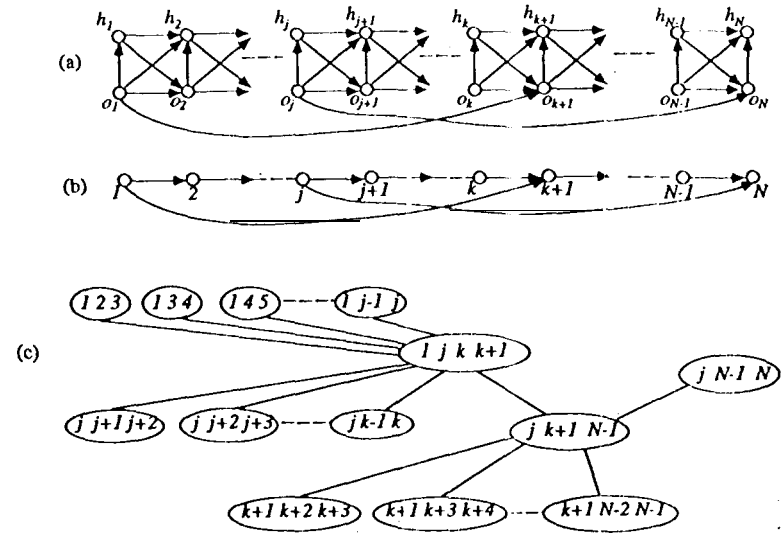

Figure 4: (a) A modified hidden Markov model for protein modeling. (b) A simplified version of this model. (c) A corresponding clique tree.

node o_N has a non-local depedency on observable node o_j. These particular non-local dependencies are complex enough that it is not obvious how to perform inference using this graph in any direct manner (e.g., by extension of the forward backward algorithms), and yet the dependencies are simple enough to permit discussion and analysis of the clique tree solution.

Figure 4b shows a simplified version of the original graph in Figure 4a. For the purposes of illustration, the hidden and output units at each time step have been collapsed into a single node at each time step. Let $m_2 = m.l$ be the number of states per node in the new graph where m is the number of states of the hidden nodes and 1 is the number of states of the observable (e.g., $1 = 4$ for DNA problems), Figure 4c shows a (non-unique) clique tree resulting from the application the JLO algorithm to the DAG of Figure 4b. For this tree, $S_N = (N-4)m_2^3 + m_2^4$. *Thus* the complexity of inference for this clique tree is $O((N+m_2)m_2^3)$ *which scales only linearly in* N, i.e., the complexity does not scale exponentially in the "distance" (k or N – j) of the non-local dependencies. Note again that this relatively simple model with only two non-local dependencies is only used for illustrative purposes: arbitrary numbers of non-local dependencies can be handled in general but the complexity of inference seems likely to scale as a function of the number of "intersecting" links.

Extensions such as these are extremely valuable in biological sequence analysis, as they can potentially capture crucial non-local information which is known to reside in 3D macromolecular structure. For example, techniques such as circular dichromism can often give information about a small number of non-local bonds between particular amino acids in a sequence. This information severely constrains the number of possible macromolecular configurations, with ensuing implications for the alignment process. It can be judiciously introduced into the alignment process through the use of dependencies such as those shown in Figure 4a. It is also possible that the even harder problem of inferring which amino acids interact strongly with each other can be approached in this way, in a manner similar to that used to investigate the process of protein folding itself. In a wider context, non-local dependencies can also be used to model the tertiary (3D) structure of RNA molecules. Standard approaches to this problem focus upon the more tractable issue of secondary structure, and "align molecules using a recursive dynamic programming algorithm. These methods, however, exclude the consideration of sophisticated 3D structures such as pseudo knots which are known to exist in RNA. A pseudo-knot is essentially a pair of overlapping non-local dependennncies in the argot of our paper.

4 Inference on Regular DAGs

In this section, we derive some specific results on the complexity of inference for certain general families of regular graphs. In all cases there is an underlying directed "chain" component to the graph consisting of N discrete $m-$ ary variables, $x_1, \ldots . x_N$, where x_i has a link pointing to x_{i+1}, $1 \leq i \leq N - 1$. We are interested in the effect of additions to this graph to model non-local dependencies. As a **convenience, we define** the complexity of inference *per node* as $C_N(G) = C(G)/N$ for graph G. Also, as we noted in the previous two sections, we may be able to construct more than one clique tree for a given DAG. Consequently, the identification of the complexity of inference for a particular clique for a given graph results only in an upper bound on inference complexity.

4.1 Hidden Versus "Non-Hidden" Models

Consider an arbitrary graph G where $C_N(G_T)$ is known, i.e., in addition to the local links, the x_i can be connected as an arbitrary DAG. Now, consider that we construct a new DAG G* such that for each xi in G, there is a directed link going from x_i to a corresponding observable variable o_i, $1< i \leq N$ (we can assume the o_i's are discrete, taking 1 values). This corresponds to a typical HMM (e.g., for speech modeling) where the x_i are "hidden" or unobserved, but now, unlike the standard first-order HMM, the underlying hidden process is not constrained to be a chain. The augmented graph G* must have C_N (G*) = $C_N(G_T) + O(ml)$ by virtue of the fact that the clique tree for G* is obtained from G_T simply by adding the clique containing $\{x_i, o_i\}$ to the graph and linking it to any clique containing x_i and repeating this process for each o_i. Thus, as long as the observable o_i only have local dependence, the basic inference proper-tics of the hidden model can be derived as extensions of those of the "non-hidden" model. Thus, the complexity remits below which are stated for "non-hidden" models are directly relevant for hidden models where each observable only depends on one hidden variable.

4.2 Locally-Bounded Dependencies

If a DAG G of "length" N can be decomposed into $\frac{N}{d}$ identical sub-graphs of "length" d, where the sub-graphs are arbitrary DAGs with complexity $C_N(G_d)$ for each, where there are no links between subgraphs except from neighboring x_i to x_{i+1}, and where the directions of all links in the sub-graphs are from "left-to-right" (for any link directed from x_j to x_k, we must have $k > j$), then $C_N(G) = C_N(G_d) + O(\frac{m^2}{d})$. Thus, certain 'decoupled" forms of DAGs satisfy a basic decomposition property in that the complexity of inference for the overall DAG scales in the same manner as the local sub-graphs plus a term to account for the propagation of information between sub-graphs.

4.3 "End-to-End" Dependencies

In addition to the underlying Markov chain, let there be an arbitrary number of directed links going from the set of nodes $x_1, \ldots , x_k$ to the set Of nodes $x_{N-k+1}, \ldots . x_N$, where $1 < k < N/2$. Thus, in the most complicated case the first k nodes can be fully connected to the last k nodes in the chain. From the resulting clique tree G_T, the sum of the clique state spaces is $S(G_T) = m^{2k} + (N - 2k)m^3$ and $C_N(G_T) = O(m^{2k}/N)$ which scales exponentially in a factor proportional to the *number* of long distance dependencies but *not in N, the length of the dependency.*

4.4 Symmetric Patterns of Dependencies

In addition to the underlying Markov chain, let there be a set of symmetric dependencies such that there is a dependency from x_i to $x_{N-i+1}, 1 \leq i \leq N/2$ (assuming N is even). By redrawing this graph in a folded manner so that x_1 appears above x_N, etc., one can easily determine the clique tree structure for the problem: the resulting inference complexity $C_N(G_T) = O(m^3)$. This is another example of a DAG structure where the dependency links in the graph do not intersect and the complexity of inference dots not depend on the number of non-local dependencies or their length.

5 Related Topics and Conclusion

Due to space constraints we can only briefly mention a number of related topics which we are currently investigating. The propagation phases of the JLO procedure (e.g., the calculation of likelihoods given observed evidence) can be carried out in a completely *local* manner suggesting parallel implementations of inference algorithms for certain applications. In addition, we are exploring the applicability of local clique tree methods to the learning problem. The results in the paper are described for discrete-valued variables: there are obvious extensions to include real-valued variables which will depend on the nature of the density functions and the dependence of the real-valued variables on other variables—for example it should be possible to include "non-parametric" black boxes such as feedforward neural networks in certain parts of the graph. Finally, it is of significant interest to try to extend this approach to undirected graphs (Markov random fields) -we have so far found that certain technical difficulties prevent a direct mapping of the methods described in this paper.

Acknowledgements

We thank Koos Rommelse for useful discussions concerning the JLO algorithm. The research described in this paper was carried out in part by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and author PS was supported in part by ARPA under grant number N00014-92-J-1 860.

References

W. Buntine, 1994, "Operations for learning with graphical models," *JAIR*.

P. Baldi, S. Brunak, Y. Chauvin, J. Engelbrecht, A. Krogh, 1994 "Hidden Markov models for human genes," Advances in Neural Information Processing Systen is 6, 761--768.

A. P. Dawid, 1992, "Applications of a general propagation algorithm for probabilistic expert systems," Statistics and Computing, 2, 25-36.

P. Frasconi and Y. Bengio, 1994, "An EM approach to grammatical inference: input/output HMMs," in Proceedings of the 12th IAPR Intl. Conf. on Pattern Recognition, IEEE Computer Society Press, 289-294.

F.V. Jensen and S.L. Lauritzen and K.G. Olesen, 1990, "Bayesian updating in recursive graphical models by local computations? Computational Statistical Quarterly, 4, 269--282.

S. L. Lauritzen and D. J. Spiegelgalter, 1989, "Local computations with probabilities on graphical structures and their application to expert systems (with discussion) ," J. Roy. Statist. Sot. Ser. B **50, 157–224.**

L. Saul and M. Jordan, 1995, "Boltzmann chains and hidden Markov models," Advances in Neural Information Processing 7, to appear.